RAID-5 DISK HAVING CACHE MEMORY IMPLEMENTED USING NON-VOLATILE RAM

PRIORITY CLAIM

[0001] This application claims priority under 35 U.S.C. §119 to Provisional Patent Application Serial No. 60/424,152 filed November 6, 2002, which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention relates to the operation of computer data storage, and more particularly, to a RAID-5 disk having cache memory and the operating method thereof.

BACKGROUND

[0003] The installation and use of cache memory in a computer system is common, since the use of cache memory can enhance computer program execution speed significantly. Further, the use of so-called "redundant array of inexpensive disks" (RAID) is commonly used to store critical data. As the price of RAID storage decreases, its applications increase and it is envisioned that RAID storage may be available to consumers for personal use.

[0004] Disk storage devices are mechanical in nature. This can be problematical insofar as a computer system accessing the disk storage faces a bottleneck. Disk input/output is generally slower than microprocessor operations and memory access. In general, a RAID system organizes a plurality of disks to create a virtual disk volume. Further, there is usually a RAID controller that accepts the input/output requests and dispatches it to the disks.

[0005] There are several RAID architectures that organize the disks in different manners to provide different advantages. For example, a "RAID-0" system distributes data blocks throughout the disks to expand the storage volume, to balance the loading of each disk, and

to enhance throughput. A "RAID-1" system provides a duplicate set of mirrored disks and stores each data block to the paired disk. Such an architecture solves the reliability problem and keeps data even after any one disk failure. The disadvantage of such a system is that it doubles the disk cost.

[0006] In yet another RAID system, known as the "RAID-5" system, the data is interleaved blockwise over all of the disks and parity blocks are added and distributed over all the disks. This provides reliability similar to a RAID-1 system and can recover data when a single disk fails by reading the parity block in other data blocks on the same stripe. On the other hand, the RAID-5 architecture provides a larger space than a RAID-1 system because it only uses one disk for the redundant data. Raid-5 systems also balance the loading of the disks by distributing parity blocks over all of the disks. One drawback of the RAID-5 system is that it generates in general more disk input/outputs for each write operation. A write operation to a block of a RAID-5 volume will be dispatched as two read operations and two write operations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Figure 1 shows in schematic form a RAID storage with a RAID controller formed in accordance with the present invention.

[0008] Figure 2 shows in schematic form a conventional RAID storage system.

[0009] Figure 3 shows in schematic form a RAID disk erase system formed in accordance with the present invention.

[0010] Figure 4 shows a sequence of disk accesses that relocate to the same disk block in accordance with the present invention.

[0011] Figure 5 shows a flow diagram illustrating a disk write request.

[0012] Figure 6 shows a flow diagram of a disk read request.

[0013] Figure 7 shows a flow diagram of a process running on a host CPU that flushes dirty blocks to the disk according to the present invention.

DETAILED DESCRIPTION

[0014] The present invention provides a computer implemented cache memory for a RAID-5 configured disk storage system to achieve a significant enhancement of the data access and write speed of the raid disk. The present invention adopts a memory cache between the RAID-5 controller and the RAID-5 disks to speed up RAID-5 system volume accesses. It utilizes the time and spatial locality property of parity blocks. The memory cache is central in its physical architecture for easy management, better utilization, and easy application to a generalized computer system. The cache blocks are indexed by their physical disk identifier to improve the cache hit ratio and cache utilization.

[0015] In addition, the present invention includes a flush daemon program for flushing dirty blocks existing in a cache memory back to the disk, a write procedure to copy data to the cache, or to allocate data block space in the cache, and to wake up the flush daemon when needed. Further, a read procedure is provided to read block data already existing in the cache memory or to allocate data block space in cache memory and then to set the data block to the cache space and read data from the disk to the cache.

[0016] Figure 1 shows an example of a computer system hardware structure that may be used to implement the present invention. The computer system consists of a host CPU 100, a memory module 110, and a plurality of disks. The disks in the computer system can be a SCSI ("scuzzy") disk 160 connected via an SCSI controller 130. Alternatively, IDE disks 150 may be connected via an IDE controller 120. Still alternatively, iSCSI disks 140 may be connected via network cards 170. The disks are configured as one or more RAID 5 disk volumes. The present invention can be applied to either a general purpose computer system in which the RAID-5 control and cache control of the present invention are software programs running on the host CPU 100. Alternatively, the present invention may be implemented as a storage subsystem that serves other computers and the host CPU 100 is dedicated to the RAID control and cache control functions of the present invention.

[0017] Figure 2 shows a block diagram of a traditional RAID system. The RAID controller 200 is either a hardwired controller or a CPU controller that processes the input/output request to the disk array. In a RAID system, a plurality of physical disks 210, 220, 230, and 240 are configured to be a virtual disk volume. The RAID controller 200 will process the

input/output request to the virtual volume and dispatch them to the physical disks 210-240. In general, the RAID system can provide higher reliability by redundant disks or better throughput by parallelizing the disk access channel. The real advantage of a RAID system depends on the RAID type.

[0018] The present invention is applied to a RAID-5 system in which the RAID-5 system uses one disk block of the physical disks as the parity block. It is the parity of the data blocks on the other disks. In a RAID-5 system, any stored data can be recovered in the case of a single disk failure by reading the parity block and the other data blocks.

[0019] Figure 3 shows in block diagram form a RAID-5 system formed in accordance with the present invention. A RAID cache 205 is added between the RAID controller 200 and the disks 210-240. In the implementation shown in Figure 1, the RAID cache 205 is equivalent to the memory module 110 of Figure 1. As noted previously, disk input/output is generally slower than CPU operations and memory accesses. The cache memory 205 is placed between the access path of the CPU and the physical disks to speed up disk access.

[0020] The RAID cache 205 is a global memory that serves all of the disks 210-240. A global memory cache is advantageous for several reasons. First, in a general computer architecture, it is easier to access a global memory cache and the cache hit transmission time is smaller because the memory bus is much faster than a disk bus. Therefore, the general computer can access the RAID cache 205 much faster than if the cache memory were distributed to the individual disks 210-240. Second, it is easier to apply battery backup memory or nonvolatile memory to the global memory cache 205 in order to maintain data consistency after a system crash. In other words, by having a single RAID cache 205, this is more reliable in general than having distributed cache memories in each of the disks 210-240. Third, the cache hit ratio and data utilization is higher in a bigger global memory cache than several small scale memory caches.

[0021] Figure 4 shows an example of a sequence of RAID-5 disk write operations that can improve RAID-5 performance in accordance with the present invention. It is also explained below why the RAID cache 205 is needed in addition to the buffer cache that is commonly used to delay disk write operations in most operating systems. In a RAID-5 system, the

data is interleaved blockwise over all of the disks and parity blocks are added and distributed over all the disks recover data when a single disk fails.

[0022] In general, a write operation to a block of a RAID-5 volume will result in the dispatch of two read operations and two write operations. The RAID controller will read the data block from a disk and read the parity block from another disk. The RAID controller will then compute the new parity block by "exclusive or" ("XOR") operations on the new data block, the old data block and the old parity block. Finally, it writes back both the data block and the parity block to the disks. The four disk operations for a disk write is one performance bottleneck in a RAID-5 system.

[0023] For example, in Figure 4, four disks 210-240 are configured as a RAID-5 volume. Two contiguous write operations write data to contiguous block i 300 and block i+1 310 of the RAID-5 volume. A normal buffer cache does not have any benefit in this general case because they are different blocks and the data in the buffer cache cannot be reused. This would require eight disk input/outputs to complete these two write operations.

[0024] With a RAID cache 205, the RAID controller 200 will dispatch a write operation to the data block and the parity block on the specific physical disks first and then look up the blocks in the RAID cache 205. The blocks in the cache memory 205 are indexed and managed by their physical disk identifier and physical block identifier. In this case, because the parity blocks 320 and 330 are the same block for contiguous data, the parity block is cached in memory after the first read. It only requires six disk input/outputs for these two operations. In general, for a RAID-5 system with N+1 disks, the number of disk input/outputs for a contiguous write will reduce to 2N+2 compared to 4N with the prior art.

[0025] In the present invention, a write-back cache policy is used for the write operation. For a write operation, data is only written to the RAID cache 205. The RAID controller 200 then updates data from the RAID cache 205 to the disks 210-240 only when necessary. To manage cache blocks in the cache memory 205 there are two flag bits for cache block. A "dirty bit" is used to indicate if a block has been modified, and it is set when data is written to the cache block and cleared when the cache block has been written to one of the disks 210-240. A "cached bit" is used to indicate if there is valid data in the RAID cache

205, and it is set when data is loaded from the disks 210-240 to the RAID cache 205 and cleared if an error occurs.

[0026] Figure 5 shows a flow diagram of the operation of writing a block to a physical disk. The RAID controller 200 accepts input/output requests to the RAID-5 volume and dispatches it to read/write requests to the data block and parity block of the physical disks. The flow diagram applies to both data blocks and parity blocks.

[0027] First, at step 400, the data from the write request is analyzed to determine if the data block has already been stored in the RAID cache 205. If it is not stored in the RAID cache 205 already, then the RAID controller 200 allocates blocks within the RAID cache 205 for storing the data at step 410. In general, the controller will allocate those cache blocks that are not "dirty" and have been least used to the data blocks. However, if the write request contains data in the RAID cache 205, then at step 420, the data is copied into the RAID cache 205. Then, at step 430, the dirty bit flag is set to dirty for that block.

[0028] The RAID controller 200 can reply to the computer or microprocessor that the write request is complete immediately and does not have to wait until the data has been written to the disks 210-240. Finally, the RAID controller 200 will check the number of dirty blocks in the RAID cache 205 at a step 440. If it is larger than a threshold, the RAID controller 200 will initiate a flushing step 450 to write the dirty blocks from the RAID cache 205 to the disks 210-240.

[0029] Figure 6 shows a flow diagram of the operation of reading data of a block from a physical disk. First, at step 500, it determines whether the reading block is already stored in the RAID cache 205. If it is not stored in the cache memory, the RAID controller 200 will allocate a not dirty and least used block to the reading block at step 510. The RAID controller 200 sets a flag of the block to "cached" at step 520 and then copies data from the physical disks 210-240 to the allocated cache block at step 530. Finally, it returns data from the RAID cache 205 and the read request is completed.

[0030] In the present invention, the RAID controller 200 does not write data to the disks and wait for a write complete on the write request. Instead, a flush thread takes the responsibility for writing dirty blocks to the disks 210-240. It will write dirty blocks to the

disks 210-240 when the number of dirty blocks or the space of dirty blocks is over a predetermined threshold.

[0031] Figure 7 shows the working flow of the flush thread method. In general, the flush thread usually sleeps at step 600. It will awaken when a predetermined time has been expired or the number of dirty blocks is over a threshold (from step 450 of Figure 5). The flush thread will check if the number of dirty blocks and the space of dirty blocks is over a threshold at step 610 and 620. The threshold values may be set to control the frequency of the flush thread operation. In general, the flush thread will also write all dirty blocks to the disks 210-240 when the system is shut down. At step 630, the dirty blocks on the RAID cache 205 will be written at step 630 and the blocks are set to "not dirty."

[0032] The RAID cache 205 can be implemented in a variety of hardware configurations. For example, in accordance with one aspect of the present invention, the RAID cache 205 is a non-volatile random access memory (NVRAM). This implementation is advantageous since data will not be lost from the RAID cache 205 during power down or other event that cuts power.

[0033] For example, Figure 8 shows how the NVRAM may be utilized in a startup environment. First, at box 800, a check is made as to whether or not the NVRAM is active. If it is not active, then the NVRAM at box 840 is reset and enabled and the startup process continues. However, if the NVRAM is active, at box 810, a check is made to see if the NVRAM configuration is set to dirty. If not, then the NVRAM continues the startup process. However, if the NVRAM is dirty, then all of the dirty blocks are flushed to the logical disk (disks 210-240) at box 820. Finally, at box 830, the NVRAM is set to "not dirty" after the flushing process.

[0034] Additionally, the NVRAM can be used to aid in start up after power loss. For example, during start up, the entire NVRAM is mapped to the logical disk. Then, dirty flags are checked to see if they are on or off. Finally, all dirty blocks are flushed to the logical disk and the dirty flag is reset to off.

[0035] Similarly, during a shut down process, the NVRAM can be used advantageously by first flushing all dirty blocks of said NVRAM to the logical disk. Then, the dirty flag is set off for the NVRAM.

[0036] From the foregoing, it will be appreciated that specific embodiments of the invention have been described herein for purposes of illustration, but that various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.